Towards Real-World Searching with Fixed-Wing Mini-UAVs

Morgan Quigley
Computer Science Department
3361 Talmage Building
Brigham Young University
Provo, UT 84602, USA
mquigley@byu.edu

Blake Barber, Steve Griffiths
Mechanical Engineering Department
435 Crabtree Building
Brigham Young University
Provo, UT 84602, USA
dbb35@email.byu.edu
stepheng@email.byu.edu

Michael A. Goodrich
Computer Science Department
3361 Talmage Building
Brigham Young University
Provo, UT 84602, USA
mike@cs.byu.edu

***Abstract*— We discuss several techniques that assist in the field deployments of fixed-wing mini-UAVs to assist search teams, focusing on automatic takeoff and landing. We also present a real-time flightpath generation routine that performs a spiral search centered around a target point, and a path planner that creates waypoints for searches up mountain canyons.**

***Index Terms*— UAV operations, automatic takeoff, automatic landing, aerial searching, flightpath generation.**

I. Introduction

Searching applications of mini-UAVs can be seen as consisting of three distinct phases: takeoff, search, and landing. In this paper, we will present our approaches to automating and simplifying these operational phases.

Though they comprise only the first and last minute of a UAV mission, takeoff and landing are often the most technically difficult maneuvers in a mini-UAV mission. Often, UAV teams will employ an expert human pilot for these phases. On larger UAV platforms, the presence of a human pilot can clearly be justified as a safety measure in the event of onboard autonomy failure. However, when dealing with small and very lightweight UAVs, reducing or eliminating the dependency on human pilots would simplify deployment of the operational team without introducing undue risk to team members. This is simply because a worst-case breakdown of UAV autonomy and loss of control is highly unlikely to have catastrophic effects to people or property on the ground due to the small size, low mass, and low velocity of mini-UAVs. This assertion is especially valid when the mini-UAV is operating over undeveloped areas, as is often the case in searching tasks.

Operating on this safety assumption, and with the goal of eliminating dependency on expert human pilots to control the mini-UAV, we present a set of techniques that permit fully autonomous takeoff, searching, and landing of small UAVs.

II. UAV Hardware

Our experimental UAV hardware is designed to be as simple and rugged as possible. The majority of our experimental mini-UAVs are 42" wing-body airframes constructed of EPP foam, which are a popular class of hobby R/C planes[1]. To keep airborne weight, drag, and complexity to a minimum, the UAVs are not equipped with landing gear and must be hand-launched.

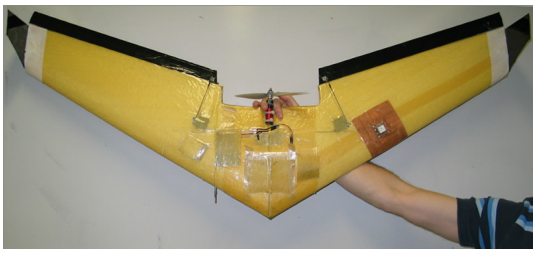


Fig. 1. A 42" wing-body mini-UAV with a push propeller

Prior work in our laboratory has created a small autopilot module based on the Rabbit 3000 microprocessor [1]. This autopilot module has since been further miniaturized and is now available commercially[2]. In addition to the autopilot and its sensors, several of our UAVs have been outfitted with a compact azimuth-elevation gimbal which carries a small color CCD camera and broadcasts live analog video to the ground station.

Prior work has also produced computationally simple algorithms that allow the autopilot to aim the camera gimbal at arbitrary target points while also generating an orbital flightpath. The design of the camera gimbal and the control algorithms for gimbal aiming and orbit generation are described in detail in a previous paper [2].

III. Automatic Takeoff

The first few seconds of flight are often the most failure-prone in hand-launched mini-UAV flights, as the aircraft must be launched at sufficient airspeed, at the proper angle of attack, with no angular velocity in any axis, and skillfully piloted. However, even with an optimal launch, the first few seconds of low-speed flight can be described as "touchy" at best, and allowing the autopilot to control this phase of flight has the potential of simplifying flight operations.

On UAVs with a pull-propeller configuration, automatic takeoff can be accomplished by simply commanding the

[1]Airframes of this type can be purchased from `www.zagi.com` and `unicornwings.site.yahoo.net`

[2]`www.procerusuav.com`

Report Documentation Page

Form Approved
OMB No. 0704-0188

1. REPORT DATE	2. REPORT TYPE	3. DATES COVERED
2005		**00-00-2005 to 00-00-2005**

4. TITLE AND SUBTITLE	5a. CONTRACT NUMBER
Towards Real-World Searching with Fixed-Wing Mini-UAVs	5b. GRANT NUMBER
	5c. PROGRAM ELEMENT NUMBER
6. AUTHOR(S)	5d. PROJECT NUMBER
	5e. TASK NUMBER
	5f. WORK UNIT NUMBER
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) **Brigham Young University,Computer Science Department,33361 Talmage Building,Provo,UT,84602**	8. PERFORMING ORGANIZATION REPORT NUMBER
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)	10. SPONSOR/MONITOR'S ACRONYM(S)
	11. SPONSOR/MONITOR'S REPORT NUMBER(S)

12. DISTRIBUTION/AVAILABILITY STATEMENT
Approved for public release; distribution unlimited

13. SUPPLEMENTARY NOTES
The original document contains color images.

14. ABSTRACT
see report

15. SUBJECT TERMS

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT	b. ABSTRACT	c. THIS PAGE			
unclassified	**unclassified**	**unclassified**		**6**	

autopilot to hold a modest climb rate, zero roll, and a medium airspeed. The autopilot then spins up the propeller while the operator is still holding the UAV, after which the UAV thrown while under full power, with flight control occurring as normal.

However, the situation is quite different for small flying-wing UAVs with a push-propeller configuration, as the hand of the operator must be in the propeller arc during launch (Figure 1). For a successful takeoff to occur, the propeller must be spun up as quickly as possible after the UAV has been thrown, as long delays between throw and spin-up result in stalling the airfoil, which renders the UAV virtually uncontrollable.

Our solution to this problem is to place a photoresistor in such a location as to be naturally covered by the operator's fingers when the operator's hand is in the propeller arc. We have also placed a piezoelectric buzzer on the airframe so that the autopilot can communicate, by various tone patterns, the state of the automatic takeoff algorithm. The photoresistor is connected in a simple resistor network to produce a varying voltage based on the amount of light striking the photoresistor. This voltage is sampled at approximately 100 Hz by onboard A/D converters.

To be able to distinguish between a true hand-launch and a mishap such as a dropped UAV, the output of the X-axis (forward-facing) accelerometer is monitored during the takeoff sequence. If the x-axis accelerometer does not measure an enormous positive acceleration, the takeoff is aborted. To determine an appropriate threshold, we repeatedly hand-launched the UAV with the motor disconnected and glided it to a landing. The maximum X-acceleration for each launch was recorded, and we then set the threshold at 90% of peak of a typical launch. If the UAV was dropped or poorly thrown, the accelerometer readings would not exceed the required threshold, and the launch would be aborted before the propeller spun up. Similarly, the differential pressure sensor on the autopilot is also used as a sentry on the launch process: if the airspeed reading is below a pre-calibrated threshold, the takeoff is aborted. In theory, one could rely on the accelerometer readings and airspeed estimates to determine the moment of launch. However, we believe that including the photoresistor is prudent due to the dangers of having the operator's hand in the propeller arc.

The automatic takeoff routine is a simple linear state machine in which the transition arcs are as follow:

1) Photoresistor senses darkness
2) Large X-acceleration is felt
3) Forward airspeed is detected
4) Photoresistor senses ambient light
5) User-programmable delay occurs
6) Propeller spins up; autopilot holds a moderate climb and slight roll to climb in a spiral around the operator.
7) Target altitude is reached; autopilot enters a holding pattern awaiting further commands.

In any state, if the conditions for transition to the next state are not met, the takeoff is aborted. Sensor readings from a normal takeoff sequence are matched with video frames that occurred at the same time in Figure 3 .

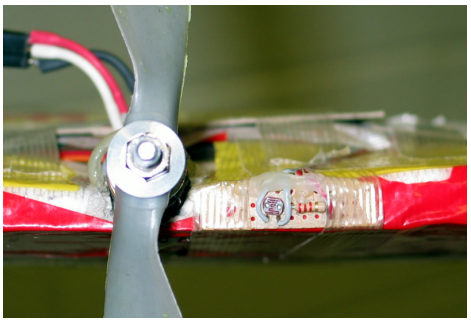


Fig. 2. Placement of the photoresistor: just to the right of the propeller hub.

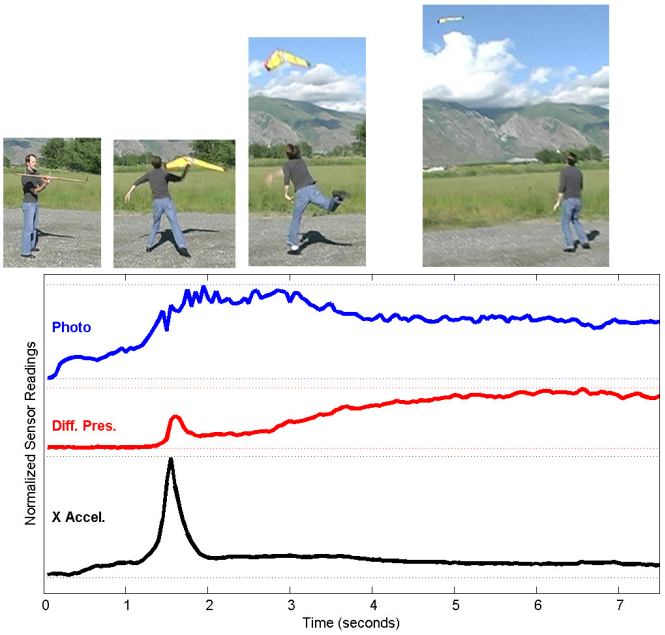


Fig. 3. Launch timeline. The photographs are extracted from a video, and the frames line up in time with their horizontal location above the plot.

If the UAV team consists of more than one person, an alternative takeoff scheme is to employ a second team member to simply flip a switch on the ground station to signal the autopilot that the UAV is airborne and it is time to fire the propeller. We often use this "two-man takeoff" technique in our field trials, but have developed the previously discussed "one-man takeoff" technique to support single-operator mini-UAV deployments.

IV. Searching

Once airborne, the UAV must perform its primary mission. Often, this involves a searching task of some form. There has been a great deal of research on robotic foraging, (e.g. [3] [4]) and searching with a UAV is conceptually a similar problem. However, UAV implementations have new constraints, such as the fact that a fixed-wing UAV cannot stop moving.

The following sections describe two very different control methods for searching with UAVs.

A. Spiral Search Patterns

Some common searching situations, such as finding victims of water accidents, involve an initial starting point for the search such as an estimate of the victim's last known position. Searchers typically begin their search at this estimated position and search the surrounding area as time allows. Moving the *camera target* of the UAV in an expanding spiral pattern is an approximation of this behavior. Because aircraft autonomy can generate real-time orbital trajectories around camera target points, all that is necessary to produce this behavior is to produce a set of equally spaced points that cover the area in a spiral pattern, and upload these camera target points in a timed sequence.

We have adapted a form of the Spiral of Theodorus [5] to derive a simple set of equations for generating spirals with constant velocity and constant ring separation:

$$r = r + \frac{\alpha\beta}{r}$$

$$\theta = \theta + \frac{\alpha}{\beta\theta}$$

$$x_{cam} = x_0 + r \cdot cos(\theta)$$

$$y_{cam} = y_0 + r \cdot sin(\theta)$$

where (r, θ) is the camera point in polar coordinates referenced from the search center, (x_0, y_0) is the center of the search, and (x_{cam}, y_{cam}) is the output of the system: the desired camera target. The parameters α and β make it possible to vary the space between the points (the search speed) and the space between the spiral arms (to compensate for camera field-of-view and UAV altitude).

A real-world flight test, conducted in calm wind conditions, is plotted in Figure 4. The camera target points were deliberately generated with a slow groundspeed and with considerable overlap of UAV orbits to show the symmetry of the automatically generated flightpath.

B. Linear Searching

Searches involving rugged mountain terrain may involve narrow mountain valleys. Aerial searching is able to cover a great deal of terrain much easier than ground-based searching in these environments [6]. However, safety considerations required for manned aircraft have led to a guideline that fixed-wing aircraft not enter mountain canyons that are too small to allow execution of a 180-degree turn [7]. Because unmanned aircraft can be constructed much smaller than manned aircraft, they are much more maneuverable and thus more able to operate in constrained environments.

Using georeferenced digital elevation maps, it is possible to generate flightpaths that take the UAV through mountain canyons that are inaccessible to low-level flights with manned aircraft. In moving towards this goal, we have created an A*-based path planner that will generate paths to a target point

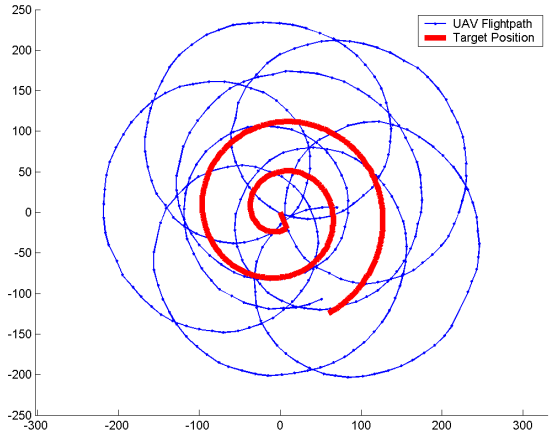


Fig. 4. Telemetry from a real-world flightpath (thin blue line) automatically generated onboard the UAV in response to the slowly spiraling camera target point (thick red line). On this flight, $\alpha = 0.006$ and $\beta = 10$. The orbital radius is 100 meters.

using a cost function that incorporates both path length and altitude changes.

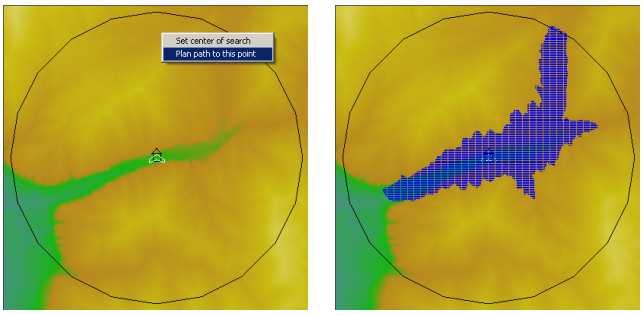
The A* planner searches for paths within a circular constraint which represents the maximum range of the UAV. In our preliminary studies, this range is set at four kilometers. The progress of the path planner is shown graphically to provide the user with feedback on the progress of the algorithm (Figure 5(a)). Once the path has been calculated, the path is smoothed by iteratively removing intermediate nodes. The smoothing algorithm removes as many nodes as possible while still keeping the resultant path within the user-defined tolerance of the optimal path (Figure 5(c)). The resultant list of GPS waypoints is uploaded to the UAV, which can then fly the requested path with a forward-facing camera.

To achieve terrain following (i.e. maintaining constant height above ground) in mountainous environments, the ground station extends the UAV trajectory forward in time for fifteen seconds, sampling the terrain the UAV will pass over. Altitude targets are regularly uploaded to the UAV so that the UAV will maintain the desired height above ground.

We have only tested this style of linear search and terrain following in simulation, as several items need to be addressed before real-world testing can occur:

- Handling communications dropouts
- Handling GPS dropouts
- Handling inclines steeper than the UAV can climb

Figure 6 graphically shows the capabilities and limitations of the A* path planner. When the UAV is flying up natural canyons, as in the beginning of the flightpath (time 0 through time 600), the UAV is able to maintain nearly constant height-above ground. However, when the flightpath takes the UAV up and over a ridgeline, as occurs from time=600 to time=700, this climb rate may not be feasible due to the power constraints of the airframe. In Figure 6, the UAV makes the climb, but with only a few meters to spare at time=675.



(a) Target specification via right-click (b) Snapshot of search progress visualization

(c) Lowest-cost path (d) Close-up to show path smoothing

Fig. 5. Four snapshots of the A* path planning interface.

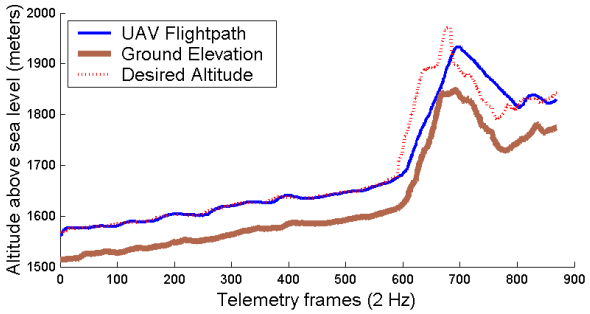


Fig. 6. Altitude versus time of a simulated UAV flying the path shown in Figure IV-B.

Although is possible to create a ground-based path planner that produces 3D waypoints and incorporates climb constraints, we believe that such an approach is overly monolithic, placing too much control on the ground station. Placing the terrain knowledge and path planning autonomy onboard the UAV would allow the UAV to safely fly back to the launch point even in the event of communications dropouts, which are bound to occur in such terrain. We are addressing this issue in current work and expect that onboard 3D flightpath generation using simplified terrain models will greatly increase the usability and reliability of the UAV in rugged mountainous environments.

C. GIS User Interface

Our current high-level interface includes several elements intended to help users quickly define search points. The interface is capable of loading elevation maps, which are readily available for most of the earth and are particularly easy to obtain with online tools provided by the United States Geological Survey[3]. Our application creates an OpenGL texture of a false-color rendering of terrain height so that subsequent redraw and zooming operations are extremely fast. All coordinates are converted to a rectangular UTM grid [8] for ease of path planning and visualization.

Road, stream, and lake data are compiled as OpenGL display lists to further increase rendering speed. This data is readily available from government agencies[4]. Figure 7 is a screenshot from this interface, showing several georeferenced aerial photos[5] mosaiced on top of a digital elevation map (DEM). The street system is drawn as white lines, and a mountain canyon extending to the right of the screenshot is seen with a stream drawn as a blue line.

Fig. 7. GIS user interface used to control high-level UAV searching behavior. Several georeferenced aerial photos are mosaiced on top of a digital elevation map (DEM). The street system is drawn as white lines, and a mountain canyon extending to the right of the screenshot is seen with a stream drawn as a blue line.

V. Automatic Landing

Landing a fixed-wing mini-UAV is not a simple task. Adding a gimbal dome to the aircraft helps make it more productive while in the searching phase of flight, but the gimbal dome raises the stall speed of the aircraft due to its disruptions to airflow, making the aircraft more difficult to land. For mini-UAVs to be useful to non-expert users, it is necessary to automate the landing sequence in order to reduce the likelihood of hardware damage in normal operations.

Automatic landing solutions have been implemented on a wide variety of fixed-wing aircraft. Very large airplanes use complex solutions that provide excellent results, but require

[3]Digital Elevation Maps are provided by the EROS Data Center of the United States Geological Survey at `http://seamless.usgs.gov`

[4]A great deal of georeferenced infrastructure information for the state of Utah can be obtained at `http://www.maps.utah.gov/` We expect that many other states and countries have similar online resources.

[5]Georeferenced aerial photography of the entire United States is available at `http://www.terraserver.microsoft.com`

more computational power and sensor precision than is available on a mini-UAV [9]. Differential GPS has been used in automatic helicopter landing routines with good results, [10] but require additional ground-based hardware for the reference receiver, and onboard hardware and software to integrate the timecode corrections.

Many military UAVs use a system involving a portable radar unit and an airborne transponder that is interrogated by the radar unit, with approaches calculated and flown by the UAV autopilot [11]. This system provides tremendous all-weather and nighttime performance, but the hardware requirements prevent it from being truly useful in our application– the aerial transponder alone currently weighs three pounds, which is more than the entire flying weight of our mini-UAVs. The existing system most related to our platform is the Pointer UAV designed and built by AeroVironment [12]. This airframe is essentially an R/C sailplane, and its autopilot performs automatic landings by executing a deep stall maneuver, where the aircraft is intentionally stalled at very low altitudes and floats down with the airfoil essentially producing drag instead of lift. This style of landing works well on airframes that are designed for it, but the stall characteristics of the flying-wing airframes in our UAV fleet are so poor that a deep-stall landing would be extremely difficult to control.

Our approach is to essentially reproduce the maneuver that experienced R/C pilots typically perform: a spiral descent to lose altitude followed by a low-speed straight glide to belly-land the aircraft. For this style of landing automation to succeed, the landing algorithm needs to have a good estimate of the height above ground (HAG) of the aircraft.

After numerous experiments and flight tests with ultrasonic rangefinders embedded in the wing of a mini-UAV, we concluded that ultrasonic HAG sensing on mini-UAVs is not reliable enough to produce consistent automatic landings over varied terrain. However, if the target application was to land on acoustically consistent surfaces such as runways or roads, ultrasonic ranging may be feasible. To allow landings to occur on natural terrain with varying foliage, our current operational approach estimates HAG through barometric pressure measurements.

A. Barometric HAG Estimation

Like many miniature autopilot modules, our autopilot incorporates a static pressure sensor to provide an altitude estimate. Before takeoff, the sensor is calibrated to reference the barometric pressure at the takeoff location as the pressure corresponding to zero altitude. Ideally, pressure (P) decreases with altitude (a):

$$P = P_{ref} - \int_{a_ref}^{a} \rho g a \, da$$

where ρ is the density of air. If ρ is sufficiently constant over the range of altitudes being measured, then altitude above the takeoff point can easily be determined:

$$HAG = \frac{P_{ref} - P_{measured}}{\rho g} = \alpha \left(P_{ref} - P_{measured} \right)$$

where α can be approximated by taking readings of the pressure sensor at a number of known reference altitudes and fitting a straight line to the data set. We have found, however, that this equation must to be modified to work in practice.

Miniature pressure sensors are sensitive to temperature, and this becomes a problem in flight operations when an autopilot is allowed to heat up as it sits during calibration and configuration. When the UAV is launched, the autopilot cools as normal airflow dissipates this heat during flight. In our experiments, this warm calibration and subsequent in-flight cooling caused the autopilots to read approximately 20 meters high. The increase in altitude readings with decreased sensor temperature was empirically determined to be linear over the relatively small range of operating temperatures. A constant of proportionality (β) was likewise empirically determined.

A further complication to the standard pressure-altitude relationship is that we do not normally equip the UAVs with static pressure tubes. Because the autopilot compartment is vented to the top of the airfoil, the speed of the airflow over the top of the airfoil causes the pressure measured along the top of the airfoil to be less than the true absolute pressure. This results in the altitude estimates computed by the autopilot to consistently read high when the UAV is in flight. The relationship between the velocity of the airflow over the airfoil and the induced offset in the static pressure reading was determined empirically through wind tunnel testing to be similar to a second order function of airspeed. This functional relationship can be combined with the linear temperature compensation term to calculate a better estimate of the true pressure:

$$\hat{P} = P_{measured} + C_1 v_a^2 + C_2 v_a + \beta \left(T_{measured} - T_{ref} \right)$$

Where v_a is the airspeed measured by the differential pressure sensor on the autopilot, and C_1 and C_2 must be determined by wind tunnel testing.

The estimated accuracy of this technique is measured by repeatedly recording the HAG estimate of the UAV at the moment of landing. Because the ground is zero meters HAG (by definition), the altitude estimate produced by the autopilot at landing is the estimate error. Using the calibration techniques described above, the UAV is consistently able to estimate the height above ground within 1.5 meters of the actual altitude relative to the home location. It is important to note, of course, that these techniques are only demonstrated to be effective within our normal operational ranges of temperature, airspeed, and very low altitudes.

B. Automatic Landing Algorithm

Using barometric HAG estimation, we are able to perform fairly accurate automatic landings on flat terrain. The algorithm incorporates user-defined approach and landing points

to generate a landing flightpath in real-time onboard the UAV. The algorithm is simple and predictable (Figure 9):

- Orbit the approach point while performing a constant-airspeed descent, using throttle only if necessary to maintain airspeed.
- When the aircraft reaches 35m HAG, break out of the far side of the orbit and glide to the landing point, using throttle only if approaching the stall speed.

Assuming the airplane control system is able to closely match the desired altitude commands, overshoot or undershoot of landing targets is solely a function of error in HAG estimation. With glide slopes of 7 to 12 degrees and measurement errors of 1.5 meters, overshoot or undershoot is expected to be 4 - 12 meters. Actual results for 20 automatic landings are shown in Figure 8. The landing target was at the origin of the coordinate system, plotted as a black square. Actual landings appear as blue asterisks. This data corresponds with the predicted error, as it shows an average over/undershoot error of 7.6 meters, with a standard deviation of 5.4 meters.

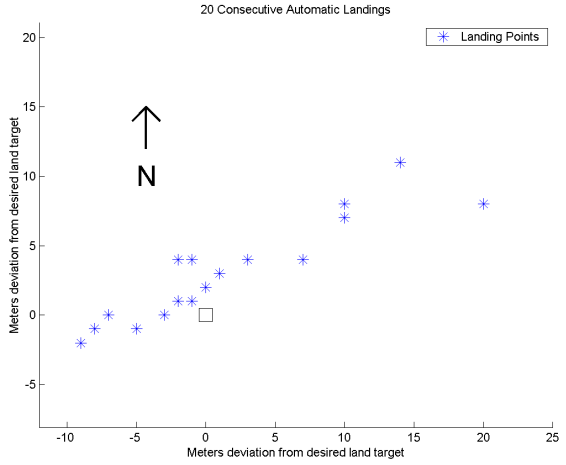
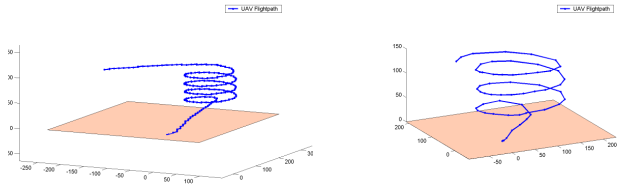


Fig. 8. Locations of 20 repeated automatic landings.



(a) Simulation (b) Real-world

Fig. 9. Auto-land trajectories in simulation (a) and telemetry from the real world (b). The UAV enters from the left, spirals down around the approach point, then exits the spiral and glides to a landing.

VI. Current and Future Work

We are currently implementing automatic flightpath generation for nonplanar environments using parallel processing onboard the airframe. The goal is to give the UAV enough autonomy that it will be able to generate spiral and linear flightpaths onboard the aircraft in spatially constrained environments such as mountain canyons. This would eliminate the current dependence on a continuous radio link with the base station, increasing UAV survivability in the event of radio shadows or failures. We are also experimenting with a variety of optical flow sensors to obtain more accurate height above ground measurements.

VII. Summary

The techniques listed in this paper present solutions to practical problems that arise in implementing real-world searching using fixed-wing mini-UAVs. Push-propeller mini-UAVs have been shown to be capable of automated takeoff and landing. Automated searching and path planning has also been shown to be feasible on these aircraft. These techniques greatly simplify field operations and show the possibility of widespread deployment of these aircraft in a variety of real-world applications.

Acknowledgment

This work was funded in part by AFOSR contract no. FA9550-04-1-0209. The authors would also like to acknowledge Ron Zeeman of Utah County Search and Rescue for practical insight about issues relating to real-world deployments.

References

[1] R. W. Beard, D. Kingston, M. Quigley, D. Snyder, R. Christiansen, W. Johnson, T. McLain, M. Goodrich, "Autonomous Vehicle Technologies for Small Fixed-Wing UAVs," Journal of Aerospace Computing, Information, and Communication, Vol 2 (2005), No.1 p.92-108.

[2] M. Quigley, M.A. Goodrich, S. Griffiths, A. Eldredge, R.W. Beard, "Target Acquisition, Localization, and Surveillance Using a Fixed-Wing Mini-UAV and Gimbaled Camera," Proceedings of the IEEE International Conference on Robotics and Automation, April 2005, Barcelona, Spain.

[3] T. Balch, "The Impact of Diversity on Performance in Multi-Robot Foraging," Autonomous Agents 1999, Seattle, WA.

[4] Zijian Ren, "Phoenix Team: A General Agent Paradigm for Rescue Simulation," Team Description for the 2004 RoboCup Rescue Simulation 2004. http://robot.cmpe.boun.edu.tr/rescue2004/

[5] P. J. Davis, Spirals: From Theodorus to Chaos, Wellesley, MA: A.K. Peters, 1993.

[6] W.G. May, Mountain Search and Rescue Techniques. Boulder, CO: Rocky Mountain Rescue Group, 1973.

[7] United States National Search and Rescue Supplement to the International Aeronautical and Maritime Search and Rescue Manual, p. 5-3 to 5-4.

[8] P. Richardus and R.K. Adler, Map Projections. Amsterdam: North-Holland Publishing Co., 1972, p.141-143.

[9] Jun Che and Degang Chen, "Automatic Landing Control using $H_{\inf}$ Control and Stable Inversion," 40th IEEE Conference on Decision and Control, Orlando, FL, December 2001.

[10] M. Bole and J. Svoboda, "Design of a reconfigurable automated landing system for VTOL unmanned air vehicles," National Aerospace and Electronics Conference 2000, Dayton, OH.

[11] A. Peterson, "Launched to Return," Unmanned Vehicles, Jan/Feb 2003. Excerpt available online at http://www.sncorp.com/media/APUV.pdf.

[12] At time of writing, a brief document describing the Pointer system is available at http://www.aerovironment.com/area-aircraft/prod-serv/ptrdes.pdf